Grant Bingham

CS 172

Prof. Tucker

Final Project Requirements

This project will accomplish two things: it will output a display of the periodic table, and it will give the user access to information on each element.

This program will have 9 classes:

- class AlkiliMetals

- class AlkilineEarth

- class TransitionMetals

- class BasicMetals

- class SemiMetals

- class NonMetals

- class Halogen

- class NobleGases

- class RareEarth

This organizes the elements on the periodic table based on their properties and their placement on the table.

Each class will look similar, and all have the same job. Each class will have a constructor that allows the user to enter the name of an element as a string argument into the constructor. Each class will also have private data members, including: element name, and properties of a given element. There will be public mutator functions that will alter the private data members based on what is entered into the constructor. There will also be accessor functions that will return the values of certain properties. Here is an example of a class.

class AlkiliMetals

{

public:

AlkiliMetals();

AlkiliMetals(string setElementName); changes string elementName

void changeProperties(); changes element properties based on what elementName is equal to.

void getLewisDotStructure(); outputs the Lewis Dot structure of an atom of the chosen element to show the number of valence electrons one atom has

string getStateOfMatter(); returns state of matter of element

int getAtomicNumber(); returns atomic number of element

double getMolecularWeight(); returns molecular weight of element

double getElectronegativity(); returns the electronegativity of element

private:

string elementName;

string stateOfMatter;

int atomicNumber;

double molecularWeight;

double electronegativity;

}:

Also, here is what a UML of this class would look like:

|  |  |
| --- | --- |
| class | AlkiliMetals |
| Data Field Members | * elementName: string * stateOfMatter: string * atomicNumber: int * molecularWeight: double * electronegativity: double |
| Behaviors | * AlkiliMetals() * AlkiliMetals(setElementName) * getLewisDotStructure() void * getStateOfMatter() string * getAtomicNumber() int * getMolecularWeight() double * getElectronegativity() double |

Now moving on to the main function. The first part of the main function will serve the purpose of displaying the periodic table. The periodic table will be made up of an array of strings called periodicElements[ ], that will contain each element abbreviation, and also borders for keeping each element separate. Since the periodic table is an irregular shape. I will have to output some of the rows of elements one at a time. The first row would look something like this for example:

cout << “| “ << periodicElements[0] << “ |” << setw(30) << “| “ << periodicElements[1] << “ | << endl;

However, I will probably use a for loop for rows that have elements from end to end. The fourth row would look something like this for example:

for(int i = 18; i < 36; i++)

{

cout << “| “ << periodicElements[i] << “ “;

}

The next part of the main function will tell the compiler to read elements from the file “elements.txt”. I will write the text file “elements.txt”, and it will be a list of all the elements on the periodic table. They will not be abbreviated. In my main function, I will create a string array called elements[ ]. As the compiler goes through the file, it will store each element as a string in elements[ ]. The classes will end up taking a specified array slot from elements[ ] as a string argument.

Lastly, I will prompt the user to enter the name of an element. Prior to the cin statement, I will declare a variable called string elementChoice. The user will enter their desired element into the string. After the user has entered their element, I am currently planning having a series of if-statements that will enter the corresponding elements[ ] slot into the appropriate class constructor, and output the element’s properties.